Shorewall and Native IPSEC

Tom Eastep
Linuxfest Northwest
April 30, 2005

©2005 – Thomas M Eastep
Disclaimers

- Although I am a Software Architect with Hewlett-Packard, Shorewall is completely my own work.
- Shorewall is not supported by HP in any way.
- My main Linux expertise is in Networking and Firewalls but I don't claim to be an IPSEC expert.
Virtual Private Networking (VPN) on Linux has followed a common model.

Prior to kernel 2.6, IPSEC followed that model, although IPSEC is not really a VPN mechanism.

In kernel 2.6, a new model for IPSEC on Linux was introduced – I'll refer to that new model as *Native IPSEC*. 
• Shorewall has had VPN support from the beginning (IPSEC support added in version 1.0.2).
• Shorewall VPN support was designed around the traditional Linux VPN model
• As a consequence, Native IPSEC is not well supported by Shorewall 2.0.* and earlier
• Support for Native IPSEC added in Shorewall 2.2.0.
Agenda

- IPSEC Overview
- IPSEC on Linux before Kernel 2.6
- IPSEC with Kernel 2.6 (Native IPSEC)
- Brief Shorewall Introduction
- Shorewall VPN Basics
- Shorewall support for Native IPSEC
- Example
- Conclusion – Q&A
IPSEC Overview
IPSEC — What is it?

- Mechanism for:
  - defining policies for secure communication in a network
  - enforcing those policies
  - defining how that communication is to be secured
  - defining how hosts authenticate themselves to each other
IPSEC Essential Reference

- Much of the information in this section of the presentation comes from there
IPSEC Basics

- Originally defined for IPV6
- “Back-ported” to IPV4
- Architecture is defined in RFC 2401
- Because it is a standard, it is widely implemented and supported
- Can be used to define complex encryption policies
- Is complex 😞
IPSEC Protocols

- Two different encapsulation protocols:
  - **Authentication Headers** (Protocol 51): Ensures datagram Integrity
  - **Encapsulated Security Payload** (Protocol 50): Ensures Datagram Integrity and optionally provides confidentiality via encryption.
- UDP port 500 (**Internet Security Association Key Management Protocol -- ISAKMP**)
- May also use IP Payload Compression Protocol (Protocol 108).
Authentication Headers (AH)

- HMAC is a cryptographic hash
- Includes IP addresses in the Header
- Hence AH cannot be used with NAT

<table>
<thead>
<tr>
<th>Next Header</th>
<th>Payload Length</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Parameter Index (SPI)</td>
<td>Sequence Number (Replay Defense)</td>
<td>Hash Message Authentication Code (HMAC)</td>
</tr>
</tbody>
</table>
Encapsulated Security Payload (ESP)

- Uses a header and a trailer around the payload
- HMAC only includes data
- Still, NAT of ESP generally isn’t possible

<table>
<thead>
<tr>
<th>Security Parameter Index (SPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
</tr>
<tr>
<td>Initialization Vector (IV)</td>
</tr>
</tbody>
</table>

Encrypted Payload

- Padding Length
- Next Header

Hash Message Authentication Code
Using AH and ESP Together

- The integrity of the IP header of an encrypted packet can be ensured by encapsulating the ESP packet inside an AH packet.

<table>
<thead>
<tr>
<th>IP Header</th>
<th>AH Header</th>
<th>ESP Header</th>
<th>Encrypted Payload</th>
<th>ESP Trailer</th>
</tr>
</thead>
</table>

13
Nat Traversal

- Routers using *Network Address Translation* (NAT) can rewrite the IP header without affecting the payload.

```plaintext
IP Header

UDP Header (Port 4500)

ESP/AH Payload
```
IPSEC Modes

- Two different modes
  - *Transport Mode* (host-to-host)
  - *Tunnel Mode* (gateway-to-gateway or gateway-to-host)
Transport Mode

- Encapsulates the payload
- The host to which the original IP packet is addressed is the host that will unencapsulate the transport mode packet
Transport Mode (continued)

- Host-to-host communication
Tunnel Mode

- Encapsulates the entire IP packet
- The host (gateway) specified in the new IP header will decapsulate
Tunnel Mode (continued)

- Clients need not run on the gateways.
- Communication between client systems and gateways is not encrypted.
Security Associations

- A *Security Association* defines how secure traffic is to be encrypted by a sending host to send to a receiving host.
- The Security Association is unidirectional.
- Security Associations are stored in a *Security Association Database* (SAD).
- Security Associations are identified by their *Security Parameter Index* (SPI).
Example (from output of ‘setkey –D’):

```
192.168.3.8 192.168.3.254
    esp mode=tunnel spi=174516471(0x0a66e8f7) reqid=0(0x00000000)
    E: blowfish-cbc 2016fe36 13490de6 1b600abd 48be832d
    A: hmac-sha1 a3c475d3 c5b180cd ad5a242d be1bd559 554334e6
    seq=0x00000000 replay=4 flags=0x00000000 state=mature
diff: 1158(s) hard: 14400(s) soft: 11520(s)
    last: Apr 9 13:10:01 2005 hard: 0(s) soft: 0(s)
current: 81962(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 817 hard: 0 soft: 0
sadb_seq=3 pid=15610 refcnt=0
```
Security Associations (continued)

- Where multiple protocols are used (e.g., AH & ESP) multiple SAs are required to form an *SA Bundle*
- Security Associations may be statically defined but are usually negotiated using a *Keying Daemon* (raccoon, isakmpd, pluto) using a method called the *Internet Key Exchange* (IKE)
- IKE uses the *Internet Security Association Key Management Protocol* (ISAKMP).
- ISAKMP uses UDP port 500 (source and destination)
Security Policies

- Define which traffic is *or is not* to be encrypted
- Define the mode (transport or tunnel)
- Define the Protocol (ESP/AH)
- Define the end-points
- Like Security Associations, Security Policies are uni-directional.
- Maintained by kernel in a *Security Policy Database* (SPD)
Example (from output of ‘setkey –DP’):

0.0.0.0/0[any] 192.168.3.8[any] any
   out ipsec
   esp/tunnel/192.168.3.254-192.168.3.8/require
lifetime: 0(s) validtime: 0(s)
spid=1369 seq=7 pid=12140
refcnt=11
IKE

- IKE defined in RFC2409
- Negotiation takes place in two Phases:
  1. Create an *ISAKMP Security Association* (ISAKMP SA), perform authentication. This phase may use either *aggressive mode* or *main mode*; main mode is preferred because it protects against “man in the middle” attacks.
  2. Create one or more IPSEC Security Associations (usually two, one in each direction) utilizing the *quick mode*. 
IKE (continued)

- ISAKMP is a framework for key exchanges.
- One widely-supported key exchange algorithm is the *OAKLEY Key Determination Protocol*.
  - Defined in RFC 2412
IPSEC on Linux before Kernel 2.6
FreeS/Wan and derivatives such as OpenS/Wan followed the traditional Linux model for VPN.
- Create a special interface
- Route VPN traffic through that interface
IPSEC before Kernel 2.6

- FreeS/Wan and derivatives
  - Created ipsec0, ipsec1, ...
  - Routing used to direct traffic to be encrypted to one of these devices
  - Unencrypted traffic goes in/out of an ipsecN interface
  - Encrypted traffic goes from local<->remote gateway
  - Required patching kernel using FreeW/Wan patches
IPSEC with Kernel 2.6 (Native IPSEC)
Native IPSEC

- Uses the BSD Model
- Kernel maintains the Security Policy Database (SPD) that defines which traffic is to be encrypted, which mode (tunnel or transport), and the end-points.
- No ipsecN interfaces are created!
- No routing involved
- Kernel maintains SAD.
- No kernel patching required (theoretically)
- Very much a work-in-progress
2.6 Kernel Patches Required

- http://shorewall.net/pub/shorewall/contrib/IPSEC
  - Patches for kernel’s 2.6.9 – 2.6.11
- http://www.netfilter.org
  - Patch-o-matic-ng “ipsec” patches (4) for kernels < 2.6.9
  - Patch-o-matic-ng “policy match” for all kernels and iptables

- Current SuSE™ 9.2 and 9.3 kernels and iptables are already patched!
ipsec-tools

- Product of the Kame project in Japan (BSD)
  - setkey – Tool for configuring SPD and for querying SPD and SAD
  - racoon – Keying Daemon supporting ISAKMP/Oakley
- Currently at version 5.1 which I recommend
- >= 5.0 is required with Kernel 2.6.11
- In Debian, ipsec-tools and racoon are separate packages!
```plaintext
● /etc/racoon/setkey.conf

# First of all flush the SAD and SPD databases

flush;
spdflush;

# Add some SPD rules

spdadd 0.0.0.0/0 192.168.3.8/32 any -P out ipsec
    esp/tunnel/192.168.3.254-192.168.3.8/require ;
spdadd 192.168.3.8/32 0.0.0.0/0 any -P in ipsec
    esp/tunnel/192.168.3.8-192.168.3.254/require ;
```
Configuring Racoon

- `/etc/racoon/racoon.conf`

```bash
path certificate "/etc/certs" ;

listen
{
  isakmp 206.124.146.176 ;
  isakmp 192.168.3.254 ;
  isakmp_natt 206.124.146.176 [4500] ;
  adminsock "/usr/local/var/racoon/racoon.sock"
    "root" "operator" 0660 ;
}
```
Configuring Racoon – Phase 1

- /etc/racoon/racoon.conf

```plaintext
# # Tipper at Home --
#
remote 192.168.3.8
{
    exchange_mode main;
    certificate_type x509 "gateway.pem" "gateway_key.pem";
    verify_cert on;
    my_identifier asn1dn;
    peers_identifier asn1dn "C=US, ST=Washington, L=Shoreline, O=Shoreline Firewall,
        CN=tipper.shorewall.net/emailAddress=postmaster@shorewall.net";
    verify_identifier on;
    lifetime time 4 hour;
    proposal {
        encryption_algorithm blowfish;
        hash_algorithm sha1;
        authentication_method rsasig;
        dh_group 2;
    }
}
```
Configuring Racoon – Phase 2

- `/etc/racoon/racoon.conf`

```conf
sainfo address 0.0.0.0/0 any address 192.168.3.8 any
{
    pfs_group 2 ;
    lifetime time 4 hour ;
    encryption_algorithm blowfish ;
    authentication_algorithm hmac_sha1, hmac_md5 ;
    compression_algorithm deflate ;
}
```
Brief Shorewall Introduction
Shorewall Basics - Zone Based

- Shorewall sees the network that it is a part of as consisting of a set of zones.
- The firewall itself comprises the zone called ‘fw’ (default value of variable $FW$).
- Zones other than $FW$ are defined in /etc/shorewall/zones.
Zone Based
(continued 2 of 4)

- Simplest model is one zone per firewall network interface. Defined in /etc/shorewall/interfaces.
- Zones are normally disjoint but may be overlapping or nested. These are defined in /etc/shorewall/hosts.
Zone Based (continued - 3 of 4)

- Shorewall assigns no meaning to zone names (1-5 characters)
- Shorewall allows you to specify a policy for connections between each pair of zones:
  - ACCEPT (allow)
  - REJECT (disallow)
  - DROP (ignore – stealth)
- Policies are defined in `/etc/shorewall/policy`
• **Rules** are exceptions to policy and are defined in `/etc/shorewall/rules`.

• **Example:**
  - *Policy*: Z1  Z2  REJECT
  - *Rule*: ACCEPT Z1 Z2 tcp telnet
Shorewall VPN Basics

- Shorewall typically runs on VPN endpoints/gateways
- Clients communicate unencrypted with the gateway.
- Gateway encrypts the traffic and sends encrypted version it to the other gateway.
- Other gateway decrypts and sends decrypted copy to receiving client.
Shorewall VPN Basics (Continued)

- Clients may run on the gateways themselves or on other computers behind the gateways.
Shorewall VPN Basics (Continued)

- Gateways must deal with two types of traffic:
  1. Unencrypted traffic between the clients.
  2. Encrypted traffic between the gateways.

- Each packet passes through Netfilter twice in each gateway!
  1. Once unencrypted
  2. Once encrypted
Shorewall VPN Basics (Continued)

- Encrypted traffic is defined using the /etc/shorewall/tunnels file.
  - Note that transport-mode encrypted traffic can still be handled in the tunnels file.
- Unencrypted traffic is handled normally (through policies and rules).
Shorewall VPN Basics (Continued)

- Columns in the tunnels file are:
  1. **TYPE** –
     - `ipsec` (AH and ESP – no NAT traversal)
     - `ipsecnat` (AH, ESP and optional NAT traversal)
     - Either may be followed by `:noah` (e.g., `ipsec:noah`) to omit rules for AH.
  2. **ZONE** – the zone that the other gateway is in.
  3. **GATEWAY** – the IP address of the other gateway (may be specified as a network or 0.0.0.0/0)

- Use of `/etc/shorewall/tunnels` is optional; see: [http://shorewall.net/VPNBasics.html](http://shorewall.net/VPNBasics.html)
Shorewall VPN Basics (continued)

- Usually define a zone for “the host(s) at the other end of the tunnel”.
- May or may not include the remote gateway
Pre-2.6 IPSEC Fit Shorewall’s Tunnel Model

- Define “host(s) at the other end of the tunnel” zone using `ipsecN` interface

```
/etc/shorewall/interfaces:

#ZONE INTERFACE OPTIONS
vpn ipsec0 ...
```

```
/etc/shorewall/hosts:

#ZONE HOSTS OPTIONS
vpn ipsec0:192.168.1.0/24 ...
```
2.6 IPSEC Does Not Fit Shorewall’s Tunnel Model

- Could define “host(s) at the other end of the tunnel” zone using `real` interface to remote gateway. This sort of works with Shorewall 2.0.*.

  /etc/shorewall/hosts:

<table>
<thead>
<tr>
<th>ZONE</th>
<th>HOSTS</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpn</td>
<td>eth0:192.168.1.0/24</td>
<td>...</td>
</tr>
</tbody>
</table>

- But we can’t guarantee that traffic to those hosts will actually be encrypted.
Shorewall Support for Native IPSEC
Defining an IPSEC zone with Shorewall 2.2.*

1. Use `/etc/shorewall/hosts` only:

<table>
<thead>
<tr>
<th>ZONE</th>
<th>HOSTS</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpn</td>
<td>eth0:192.168.1.0/24</td>
<td>ipsec,...</td>
</tr>
</tbody>
</table>

- For unencrypted traffic outbound on eth0 to 192.168.1.0/24:
  - Traffic that *will be encrypted using IPSEC* is going to zone `vpn`.

- For unencrypted traffic from 192.168.1.0/24 received on eth0:
  - Traffic that *was unencrypted using IPSEC* is coming from zone `vpn`.
2. Use `/etc/shorewall/ipsec` and `/etc/shorewall/interfaces`

**/etc/shorewall/ipsec:**

<table>
<thead>
<tr>
<th>ZONE</th>
<th>IPSEC OPTIONS</th>
<th>IN OPTIONS</th>
<th>OUT OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpn</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**/etc/shorewall/interfaces**

<table>
<thead>
<tr>
<th>ZONE</th>
<th>INTERFACE</th>
<th>BROADCAST</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpn</td>
<td>eth0</td>
<td>192.168.1.255</td>
<td></td>
</tr>
</tbody>
</table>
2. Use /etc/shorewall/ipsec and /etc/shorewall/interfaces (continued)

– For unencrypted traffic outbound on eth0:
  • Traffic that *will be encrypted using IPSEC* is going to zone **vpn**.

– For unencrypted traffic from received on eth0:
  • Traffic that *was unencrypted using IPSEC* is coming from zone **vpn**.

– The vpn zone is said to be an *IPSEC zone*. 
3. Use /etc/shorewall/ipsec with /etc/shorewall/hosts.

We will see an example later in the presentation.
Problem – TCP MSS

- MTU Path discovery (RFC 1191) depends on delivery of “fragmentation needed” ICMP packets and uses the MSS option in TCP SYN and SYN,ACK packets.
- Naïve network administrators believe that ICMP == evil and block it.
- Result: Communication problems
  - Small messages are OK
  - Large messages are dropped
TCP MSS (continued)

- Most commonly is a problem when the some sort of encapsulation occurs. PPPoE is an example.
- Also occurs in IPSEC
- Usual workaround is to use the Shorewall CLAMPMSS option in shorewall.conf.
- CLAMPMSS=yes causes the MSS option to be set to the minimum of its value and MTU minus 40.
• CLAMPMSS=<number> causes the MSS option to be set to the minimum of its current value and <number>
TCP MSS (continued)

- But in the case of Kernel 2.6 IPSEC, the device MTU doesn’t reflect the length of the IPSEC headers.
- Hence CLAMPMSS=Yes isn’t effective and CLAMPMSS=<number> affects ALL TCP connections.
- Solution: Use the ‘mss=‘ option in /etc/shorewall/ipsec.
Example
My Network
Security Policies

- /etc/racoon/setkey.conf

  # First of all flush the SAD and SPD databases
  flush;
  spdflush;

  # Add some SPD rules for Tipper
  spdadd 0.0.0.0/0 192.168.3.8/32 any -P out ipsec
      esp/tunnel/192.168.3.254-192.168.3.8/require ;
  spdadd 192.168.3.8/32 0.0.0.0/0/0 any -P in  ipsec
      esp/tunnel/192.168.3.8-192.168.3.254/require ;

- I don’t specify a policy for the XP box (couldn’t get it to work)
Racoon

- `/etc/racoon/racoon.conf`

```
path certificate "/etc/certs" ;

listen
{
    isakmp 206.124.146.176 ;
    isakmp 192.168.3.254 ;
    isakmp_natt 206.124.146.176 [4500] ;
    adminsock "/usr/local/var/racoon/racoon.sock" "root"
        "operator" 0660 ;
}
```
Racoon – Phase 1
Tipper Wireless

• /etc/racoon/racoon.conf

# Tipper at Home --
#
remote 192.168.3.8
{
  exchange_mode main ;
  certificate_type x509 "gateway.pem" "gateway_key.pem" ;
  verify_cert on ;
  my_identifier asn1dn ;
  peers_identifier asn1dn "C=US, ST=Washington, L=Shoreline, O=Shoreline Firewall,
  CN=tipper.shorewall.net/emailAddress=postmaster@shorewall.net" ;
  verify_identifier on ;
  lifetime time 4 hour ;
  proposal {
    encryption_algorithm blowfish ;
    hash_algorithm sha1 ;
    authentication_method rsasig ;
    dh_group 2 ;
  }
}
Racoon – Phase 2
Tipper Wireless

• /etc/racoon/racoon.conf

```
sainfo address 0.0.0.0/0 any address 192.168.3.8 any
{
  pfs_group 2 ;
  lifetime time 4 hour ;
  encryption_algorithm blowfish ;
  authentication_algorithm hmac_sha1, hmac_md5 ;
  compression_algorithm deflate ;
}
```
/etc/racoon/racoon.conf

# Work Laptop at Home -- 3des is the best alternative that XP supports.
remote 192.168.3.6 inherit 192.168.3.8
{
    passive on ;
    proposal_check obey ;
    peers_identifier asn1dn "C=US, ST=Washington, L=Shoreline,
    O=Shoreline Firewall,
    CN=eastepnc6000.americas.cpqcorp.net/emailAddress=tom.eastep@hp.com"
    ;
    generate_policy on ;
    lifetime time 4 hour ;
    proposal {
        encryption_algorithm 3des ;
        hash_algorithm shal ;
        authentication_method rsasig ;
        dh_group 2 ;
    }
}
Racoon – Phase 2
Work Laptop Wireless
Windows™ XP

- `/etc/racoon/racoon.conf`

```
sainfo address 0.0.0.0/0 any address 192.168.3.6 any
{
    pfs_group 2 ;
    lifetime time 4 hour ;
    encryption_algorithm 3des ;
    authentication_algorithm hmac_sha1, hmac_md5 ;
    compression_algorithm deflate ;
}
```
Both systems on the road -- We use 3des for phase I to accommodate XP. Since we don't know the IP address of the remote host ahead of time, we must use "anonymous".  

remote anonymous inherit 192.168.3.6  
{  
nat_traversal on ;  
ike_frag on;  
}
Racoon – Phase 2
Roadwarriors

- /etc/racoon/racoon.conf
  sainfo anonymous
  {
    pfs_group 2;
    lifetime time 4 hour ;
    encryption_algorithm blowfish, 3des;
    authentication_algorithm hmac_sha1, hmac_md5 ;
    compression_algorithm deflate ;
  }
#ZONE | DISPLAY | COMMENTS
---|---|---
sec | Secure | IPSEC Secure Zone
#ZONE | IPSEC | OPTIONS | IN | OPTIONS | OUT | OPTIONS
--- | --- | --- | --- | --- | --- | ---
# | ONLY | | | | | |
sec | Yes | mode=tunnel | mss=1400 | | | |
Shorewall
/etc/shorewall/interfaces

<table>
<thead>
<tr>
<th>ZONE</th>
<th>INTERFACE</th>
<th>BROADCAST</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>net</td>
<td>$EXT_IF</td>
<td>-</td>
<td>...</td>
</tr>
<tr>
<td>Wifi</td>
<td>$WIFI_IF</td>
<td>-</td>
<td>dhcp,maclist</td>
</tr>
</tbody>
</table>

- Note that I use MAC verification on my wireless network.
#ZONE  HOST(S)  OPTIONS
sec  $WIFI_IF:192.168.3.0/24
sec  $EXT_IF:0.0.0.0/0

• Note that since the sec zone is defined as an IPSEC zone in /etc/shorewall/ipsec, I don't have to specify ipsec in the OPTIONS column.
Conclusion – Q&A
Where to Find More Information

- http://www.ipsec-howto.org/
- http://ipsec-tools.sourceforge.net/
- http://shorewall.net
  - http://shorewall.net/IPSEC-2.6.html
  - http://shorewall.net/IPSEC.htm
  - http://shorewall.net/myfiles.htm
  - http://shorewall.net/VPNBasics.html